UNITED STATES PATENT APPLICATION

for

METHOD AND APPARATUS FOR NOP FOLDING

Inventors:

Jeffrey Rupley II
Edward A. Brekelbaum
Edward T. Grochowski

Attorney's Docket No.: 042390.P15757

METHOD AND APPARATUS FOR NOP FOLDING

Background

Technical Field

[0001] The present disclosure relates generally to information processing systems and,

5 more specifically, to folding no-operation instruction ("NOP") information into buffer entries for

other instructions.

Background Art

[0002] Microprocessors often use instruction pipelining to increase instruction throughput.

An instruction pipeline processes several instructions through different stages of instruction

10 execution concurrently, using an assembly line-type approach. These instructions may be

executed in a dynamically scheduled (*e.g.*, out-of-order) processor. For instructions that are

allowed to execute out of order, the instructions are retired in their original program order. Until

retirement, information regarding instructions executed out of order is maintained in a structure

· such as, for instance, a re-order buffer ("ROB"). In-order retirement of instructions that have

15 been executed out of order allows for precise exception handling.

[0003] Some processors, such as the Itanium® and Itanium II® microprocessors available

from Intel Corporation in Santa Clara, California, utilize Explicitly Parallel Instruction

Computing (EPIC) technology to execute multiple instructions simultaneously in order to

increase instruction throughput. In such processors, several instructions (*e.g.*, three instructions)

20 are grouped together into aligned containers called bundles. Each bundle includes three 41-bit

instructions and a format code.

[0004]   If a series of instructions to be executed by the processor does not fit into one of the

templates indicated by a given format code, then a no-operation instruction (referred to herein as

"NOP instruction" or simply "NOP") may be inserted into a bundle in order to execute one or

two other instructions that do fit into the template. NOP instructions may also be inserted into a

5   bundle for branch alignment reasons. If a relatively large percentage of NOP instructions appear

in the code stream, computing resources may be inefficiently utilized.

Brief Description of the Drawings

[0005]   The present invention may be understood with reference to the following drawings

10   in which like elements are indicated by like numbers. These drawings are not intended to be

limiting but are instead provided to illustrate selected embodiments of a method and apparatus

for NOP folding.

[0006]   Fig. 1 is a block diagram illustrating at least one embodiment of a processor that

may utilize disclosed techniques.

15   [0007]   Fig. 2 is a block diagram further illustrating at least one embodiment of an

instruction queue and at least one embodiment of a buffer capable of holding folded NOP data.

[0008]   Figs. 3 and 4 are flowcharts illustrating at least one embodiment of a method for

folding NOP information into buffer entries for other instructions.

[0009]   Fig. 5 is a block diagram of at least one embodiment of a processing system that

20   may perform NOP folding.

Detailed Description

[00010]   Described herein are selected embodiments of an apparatus and method related to

NOP folding. NOP folding refers to a method of maintaining information regarding a target

NOP instruction in a buffer entry associated with an instruction other than the target NOP

5   instruction.

[00011]   In the following description, numerous specific details such as processor types,

register lengths, and instruction formats have been set forth to provide a more thorough

understanding of the present invention. It will be appreciated, however, by one skilled in the art

that the invention may be practiced without such specific details. Additionally, some well

10   known structures, circuits, and the like have not been shown in detail to avoid unnecessarily

obscuring the present invention.

[00012]   As is stated above, if a large number of NOP instructions are inserted into the

instruction stream, certain resource utilization inefficiencies may result. For out-of-order

("OOO") processors, it is possible to free up some machine resources by declining to process

15   NOP instructions. However, in such cases NOP instructions are still fetched and decoded.

[00013]   Even if NOP instructions are not executed, they are nonetheless retired in program

order. In order to facilitate in-order retirement, a standard component of many OOO processor

microarchitectures is a re-order buffer ("ROB"). Placing a large number of NOP instructions

into the ROB, even if the NOP instructions are not executed, creates inefficiencies by allocating

20   ROB resources for unexecuted instructions. Embodiments of the method and apparatus

disclosed herein address this and other concerns related to processing of NOP instructions.

[00014]   Fig. 1 is a block diagram illustrating at least one embodiment of an out-of-order

processor 100. The processor 100 includes an instruction storage area 102, such as an L1 cache,

for storing instructions. The processor 100 also includes a fetch engine 104 to fetch instructions

from the storage area 102. The fetch engine 104 operates to retrieve a bundle from the storage

5   area 102 and forward it to an instruction queue 106.

[00015]   The instruction queue 106 is a memory storage structure that can hold one or more

bundles 108a-108x. For at least one embodiment, the instruction queue 106 can hold up to eight

(8) bundles. The format code in the template field of a bundle (described below) indicates which

type of execution unit may process each instruction in the bundle. From the instruction queue,

10   each non-NOP instruction in a bundle is forwarded to the appropriate execution unit 50 for

execution.

[00016]   As used herein, the term "bundle" in intended to mean a grouping of instructions.

One skilled in the art will recognize that the number of instructions grouped together to form a

bundle is not limited to three, and may be any number, including two and four. One of skill in

15   the art will also recognize that a grouping of instructions may be considered a bundle, even if the

grouping is known by another term, such as "very long instruction word" (VLIW).

[00017]   For at least one embodiment, as non-NOP instructions are issued to their respective

execution units 50a-50n, they are assigned entries in the ROB 112 for any results they may

generate. That is, a place is reserved in the ROB 112, which may be implemented as a queue, for

20   each non-NOP instruction. For at least one embodiment, such allocation is performed by folding

logic 116. Folding logic 116 need not necessarily be stand-alone logic as illustrated in Fig. 1.

For instance, for at least one embodiment folding logic 116 is implemented as part of a decoder

(not shown).

[00018]   The logical order of instructions is maintained within the ROB 112. For at least one embodiment, the assigned entry in the ROB 112 will ultimately be filled in by a value as instruction execution proceeds. The value represents the result of the instruction. For at least one embodiment, when entries reach the head of the reorder buffer 112, provided they've been filled

5   in with their actual intended result, they are removed, and each value is written to its intended architectural register (not shown). After the contents of a ROB 112 entry are written to a register, the instruction is retired by retirement logic 114.

[00019]   The number of available entries in the ROB 112 may be limited. For at least one embodiment, the ROB 112 contains 256 entries. Declining to allocate an entry in the ROB 112

10   for NOP instructions allows the entries of the ROB 112 to be allocated for operational instructions. However, even when NOP instructions are not scheduled or executed, it may not be desirable to decline to track NOP instructions in the ROB 112. For instance, it may be useful to track NOP instructions in the ROB 112 for exception-reporting purposes since a NOP may catch an exception from a prior instruction. In addition, for performance-monitoring purposes it may

15   be desirable for performance-monitoring logic to accurately count the number of retired instructions (including NOP instructions).

[00020]   Fig. 1 illustrates that, in order to avoid inefficient utilization of execution units 110a-110n and ROB 112 entries, NOP instructions are not forwarded to the execution units 110a-110n for execution. Similarly, for most NOP instructions no entry in the ROB 112 is

20   allocated. Instead, "folded" information regarding unexecuted NOP instructions is forwarded to the ROB 112 via folding logic 116. In this manner in-order NOP retirement is accommodated while conserving space in the ROB 112. Operation of the folding logic 116 is discussed in further detail below.

[00021]   Fig. 2 illustrates at least one embodiment of a bundle format 230. For at least one

embodiment, each bundle includes three 41-bit instruction slots 232, 234, 236 and a 5-bit

template field 238. The value in the template field 238 specifies the mapping of the instruction

slots to execution unit types.

5   [00022]   Fig. 2 also illustrates that the bundles 108a-108x of the instruction queue 106 follow

the bundle format 230. Three 41-bit instructions are grouped together into bundles that each

include a format code. One skilled in the art will recognize that other formats may be utilized for

a bundle. For example four instructions, or any other number of instructions, may be included in

a bundle. Furthermore, each instruction may be of any size, such as 16 bits, 32 bits or any other

10   length.

[00023]   The instructions in each instruction slot of a bundle are referred to as "syllables."

Within a bundle, execution order proceeds from syllable 0 to syllable 2.

[00024]   Fig. 2 illustrates at least one embodiment of a buffer 210 whose entries 222a-222y

are capable of holding folded NOP data. The entries of the buffer 210 illustrate bundle-level

15   NOP folding. Each entry 222 of the buffer 210 includes an information field 220 for a syllable

of a particular bundle as well as two NOP indicator fields 212, 214, one for each of the other

syllables of the particular bundle. For an embodiment that implements buffer 210 as a ROB

(such as, e.g., ROB 112 in Fig. 1), the information field 220 holds result information from the

syllable instruction.

20   [00025]   Reference is made to Fig. 2 for further discussion of fields in a buffer entry 222 for

an instruction from a three-instruction bundle. If the information field 220a is allocated for

syllable 0 of a bundle x, then the value in the first NOP indicator field 212a specifies the NOP

status for syllable 1 of bundle x. Similarly, the value in the second NOP indicator field 214a

specifies the NOP status for syllable 2 of bundle x. Of course, for at least one alternative embodiment the first NOP indicator 212 could specify the NOP status for syllable 2 and the second NOP indicator 214 can specify the NOP status for syllable 1. Also, in alternative embodiments that include a larger number of instructions (such as four or more), the entries 222a-222n include additional NOP indicator bits.

[00026]   Accordingly, the number of NOP indicator fields in a buffer entry may be driven by the number instructions defined for a bundle. For at least one embodiment, if a bundle includes n instructions, then a buffer entry includes n-1 NOP indicator fields in order to specify whether every other instruction in the bundle is a NOP. For a processor that utilizes two-instruction bundles, for example, only one NOP indicator bit is included in the buffer entry 222.

[00027]   Table 1 illustrates at least one embodiment of the mapping of information specified by the contents of a buffer entry 222 for an instruction from a three-instruction bundle:

Table 1

If contents of 220 is …	…then contents of 212 is NOP indicator for	…and the contents of 214 is NOP indicator for
Syllable 0 information	Syllable 1	Syllable 2
Syllable 1 information	Syllable 0	Syllable 2
Syllable 2 information	Syllable 0	Syllable 1

The NOP indicator fields thus map to specific instruction slots, depending on the location (within the bundle) of the instruction for which the information field 220 has been allocated.

[00028]   One will note that a potential double-marking problem exists when a single NOP appears in a bundle. For example, consider two sample bundles x and y that contain the following three instructions in their respective syllables 0, 1 and 2: x = {add, NOP, sub} and y = {add, sub, NOP}. Under the scheme illustrated in **Table 1**, the NOP in x might be recorded twice: once in field 212 for the add instruction and once in field 214 for the subtract instruction. Similarly, the NOP in y might be recorded twice: once in field 214 for the add instruction and once in field 214 for the subtract instruction.

[00029]   To avoid such double-marking, a priority rule may be imposed on the setting of values for the NOP indicator fields 212, 214. For at least one embodiment, a rule is imposed such that a NOP in a bundle containing only one NOP instruction is recorded only in the entry 222 for the lowest-number syllable adjacent to the NOP. For the x={add, NOP, sub} example, the NOP is thus specified in NOP indicator field 212 for the add instruction (syllable 0) but is not specified in the entry 222 for the subtract instruction (syllable 2). Similarly, for the y={add, sub, NOP} example, the NOP is specified in NOP indicator field 214 for the subtract instruction. The NOP is not specified in a NOP indicator for the add instruction because the add instruction is not adjacent to the NOP.

[00030]   It is important to note that a bundle may include only NOP instructions (*i.e.*, {NOP, NOP, NOP}). In such cases, a rule may be applied such that only two of the three NOP instructions are folded. For example, the remaining NOP may be allocated a buffer entry 222, with each NOP indicator field 212, 214 reflecting the other two NOP instructions of the bundle.

[00031]   For at least one embodiment, the NOP folding scheme discussed herein is an optional feature that may be disabled. Such flexibility is useful, for instance, when the processor is in single-step mode or when the processor has instruction breakpointing enabled. Turning off

the NOP folding feature during such times avoids the potential problems with folding an instruction that should be executed in order to generate an exception.

[00032] Alternatively, for at least one embodiment NOP folding is not capable of being disabled. Instead, for each folded NOP instruction an additional indicator is maintained to specify whether any potential exceptions are indicated for the folded NOP instruction.

[00033] Table 2 illustrates at least one embodiment of a scheme for folding NOP data into buffer entries (such as 222, Fig. 2) for each of several possible bundle configurations. Buffer entries may be generated by folding logic 116 (Fig. 1) according to the scheme illustrated in Table 2. In Table 2, S0, S1 and S2 denote bundle syllables that contain an instruction other than a NOP instruction. The notation [Sx] indicates that a buffer entry is allocated for results of the instruction indicated in syllable Sx of a bundle. Values in parenthesis indicate values for the NOP indicator fields 212, 214, respectively. For at least one embodiment, a logic-high value (such as 1b'1') in a NOP indicator field represents a "present" value to denote existence of a NOP instruction in the bundle slot as specified in Table 1.

Table 2

Bundle	Buffer entry for Syllable 0	Buffer entry for Syllable 1	Buffer entry for syllable 2
S0, S1, S2	[S0](0,0)	[S1](0,0)	[S2](0,0)
S0, S1, NOP	[S0](0,0)	[S1](0,1)	none
S0, NOP, S2	[S0](1,0)	none	[S2](0,0)
NOP, S1, S2	none	[S1](1,0)	[S2](0,0)
S0, NOP, NOP	[S0](1,1)	none	none
NOP, S1, NOP	none	[S1](1,1)	none
NOP, NOP, S2	none	none	[S2](1,1)

NOP, NOP, NOP	[S0](1,1)	none	none

[00034] Consider the following example. A series of three consecutive bundles in an instruction queue (such as 106, Fig. 1) contain the following instructions:

```
    ldf, NOP, fadd
5   NOP,fmpy,add
    NOP,NOP,br
```

For a processor that places each instruction, including NOP instructions, in a ROB or other buffer, the preceding series of instructions would generate nine (9) buffer entries. For a

10   processor that utilizes an embodiment of the NOP folding techniques described herein, the following buffer entries 222 are generated by folding logic 116:

```
     ldf (1,0)
     fadd (0,0)
15   fmpy (1,0)
     add (0,0)
     br (1,1)
```

Accordingly, only five (5) buffer entries 222 are generated, with the additional 4 NOP

20   instructions being "folded" into the buffer entries for the other instructions.

[00035] One skilled in the art will recognize that buffer 210 may be any structure that holds instruction-specific data, such as a ROB (112, Fig. 1). For embodiments where the buffer 210 is implemented as a ROB, the information field 220 is allocated for results of the instruction. A NOP folding scheme using NOP indicator bits (such as 212 and 214 in Fig. 2) may be used with

25   other types of buffers as well. For example, for at least one alternative embodiment buffer 210 may be implemented as a trace cache for instruction fetching. In such embodiment, the information field 220 of the buffer may contain decoded instructions or may contain micro-operations.

[00036]   Figs. 3 and 4 are flowcharts illustrating at least one embodiment of a method 300 for folding NOP information into buffer entries for other instructions. For at least one embodiment, the method 300 is performed by folding logic, such as folding logic 116 illustrated in Fig. 1.

5   While the embodiment 300 illustrated in Figs. 3 and 4 performs folding for a three-instruction bundle, one of skill in the art will recognize that the method may easily be modified to perform NOP folding for bundles of other lengths. One of skill in the art will further recognize that the blocks of Figs. 3 and 4 may be performed in a different order without straying from the illustrated functionality, and that one or more blocks may be implemented with logic that

10   performs multiple blocks simultaneously.

[00037]   Fig. 3 illustrates that the method 300 begins at block 302. Processing proceeds in order to determine the contents of a bundle (such as, for example, an entry of an instruction queue such as 106 in Fig. 1) and to allocate buffer 210 (Fig. 2) entries. It is assumed that the allocated entry includes some indicator to specify for which syllable in a bundle the entry has

15   been allocated.

[00038]   Figs. 2, 3 and 4 are referenced to discuss the method 300. The method 300, as is discussed in further detail below, determines the number of NOPs in a bundle, allocates buffer entries 222, and sets values for NOP indicator bits 212, 214. If none of the instructions in a bundle are NOP instructions, a buffer entry 222 is allocated for each instruction in the bundle. If

20   all of the instructions in a bundle are NOP instructions, an entry 222 is allocated for one of the NOP instructions and the other two NOP instructions are "folded" into that entry 222. If the number of NOP entries is greater than zero but less than the number of instructions in a bundle, then a buffer entry 210 is allocated for each non-NOP entry and the NOP instruction(s) is folded

into a non-NOP entry. As used herein, "allocating a buffer entry" for an instruction is intended to indicate that the instruction information field 220 of the allocated entry 222 is allocated for results, if any, of the instruction.

[00039]   At block 304 it is determined whether the first syllable of a bundle indicates a NOP instruction. If so, processing proceeds to block 312 to determine if the second syllable of the bundle indicates a NOP instruction. If so, processing proceeds to block 322 to determine if the third syllable of the bundle indicates a NOP instruction. If so, then it has been determined that all three instructions in the bundle are NOP instructions. As is stated above, it may be desirable in such cases to allocate an entry in the buffer 210 for one of the NOP instructions. Accordingly, at block 330 an entry 222 is allocated in the buffer 210 for a NOP instruction in the bundle. Then, the allocated entry's 222 NOP indicator bits 212, 214 are set with a "present" value to indicate that both of the other instructions in the bundle are NOP instructions. Processing ends at block 332.

[00040]   If it is determined at block 312 that the second syllable of the bundle is not a NOP, then processing proceeds to block 314 to determine if the third syllable of the bundle is a NOP. If so, then it is determined that the bundle contains two NOP instructions in the format {NOP,S1,NOP}. Processing proceeds to A in Figure 4.

[00041]   Brief reference to Fig. 4, along with Fig. 2, illustrates that at block 402 an entry is allocated in the buffer 210 for the second syllable (S1) of the bundle. At block 402, NOP indicator fields 212, 214 are set with a "present" value to indicate that both of the other syllables in the bundle (besides second syllable S1) contains a NOP instruction. Processing proceeds to B in Fig. 3 and ends at block 332.

[00042]   If it is determined at block 314 that the third syllable of the bundle is not a NOP instruction, then it is determined that the bundle contains one NOP instruction in the format {NOP,S1,S2}. Processing proceeds to block 324, wherein an entry is allocated in the buffer 210 for each of the non-NOP instructions. A first entry, entry x, is allocated for the first non-NOP instruction in syllable S1. The first NOP indicator field 212 for entry x is set with a "present" value to indicate that the instruction in syllable S0 is a NOP instruction. The second NOP indicator field for entry x is set to a "not present" value. A second entry, entry x+1, is allocated in the buffer 210 for the second non-NOP instruction in syllable S2. Neither of the NOP indicator fields 212, 214 for entry x+1 are set with a "present" value. Processing ends at block 332.

[00043]   If it is determined at block 322 that the third syllable is not a NOP instruction, then it is determined that the bundle contains two NOP instructions in the format {NOP, NOP, S2}. Processing proceeds to block 328, wherein an entry is allocated in the buffer 210 for the non-NOP instruction in syllable S2. Both NOP indicator field 212, 214 for the entry are set with a "present" value to indicate that the instructions in the remaining two syllables are NOP instructions. Processing ends at block 332.

[00044]   If it is determined at block 304 that the first syllable of the bundle does not indicate a NOP instruction, then processing proceeds to block 306 to determine if the second syllable of the bundle indicates a NOP instruction. If so, then processing proceeds to block 316 to determine if the third syllable of the bundle indicates a NOP instruction. If not, then it is determined that the bundle contains one NOP instruction in the format {S0, NOP, S2} and processing proceeds to block 318. At block 318 an entry is allocated for each non-NOP instruction in the bundle. A first entry, entry x, is allocated for the first non-NOP instruction in

-14-

syllable S0. The first NOP indicator field 212 is set with a "present" value to indicate that the instruction in syllable S1 is a NOP instruction. The second NOP indicator field for entry x is set to a "not present" value. A second entry, entry x+1, is allocated in the buffer 210 for the second non-NOP instruction in syllable S2. Neither of the NOP indicator fields 212, 214 for entry x+1 are set with a "present" value (assuming use of the rule that a NOP will be indicated in the entry for a syllable adjacent to the NOP). Processing ends at block 332.

[00045]   If it is determined at block 316 that the third syllable of the bundle does indicate a NOP instruction, then the bundle includes two NOP instructions and one non-NOP instruction in the format {S0, NOP, NOP} and processing proceeds to block 326. At block 326 an entry in the buffer 210 is allocated for the non-NOP instruction in syllable S0. The first and second NOP indicator fields 212, 214 are set with a "present" value to indicate the two NOP instructions in the bundle. Processing ends at block 332.

[00046]   If the second-syllable NOP check at block 306 evaluates to false, then processing proceeds to block 308 to determine if the bundle has a NOP in its third syllable. If not, then the bundle contains no NOP instructions and processing proceeds to block 310. At block 310, three entries in the buffer 210 are allocated, one for each instruction in the bundle. Each NOP indicator field 212, 214 for each of the allocated entries is set to a "not present" value to indicate the absence of NOP instructions in the bundle. Processing ends at block 332.

[00047]   If the third-syllable NOP check at block 308 evaluates to "true," then the bundle includes one NOP instruction in the format {S0, S1, NOP}, and processing proceeds to block 320. At block 320, two entries are allocated in the buffer 210, one for each non-NOP instruction in the bundle. A first entry, entry x, is allocated for the first non-NOP instruction in syllable S0. The first and second NOP indicator fields 212, 214 for entry x are set with a "not present" value

(assuming use of the rule that a NOP will be indicated in the entry for a syllable adjacent to the

NOP). A second entry, entry x+1, is allocated in the buffer 210 for the second non-NOP

instruction in syllable S2. The first NOP indicator field 212 for entry x+1 is set with a "not

present" value. The second NOP indicator field 214 for entry x+1 is set with a "present" value to

5   indicate the NOP instruction in syllable S2. Processing ends at block 332.

[00048]   One skilled in the art will recognize that, depending on the initialization scheme

employed for the NOP indicator fields 212, 214, some values may be preset and need not be set

as indicated in Figs. 3 and 4. For instance, if the values of the NOP indicator field 212, 214 for

each buffer entry 222 is preset to zero, then at blocks 318, 320, 310 and 402, "not present" values

10   need not be written to the NOP indicator fields 212, 214 (assuming that a logic-low value

represents a "not present" indication).

[00049]   Fig. 5 illustrates at least one embodiment of a processing system 500 that may

utilize disclosed techniques. System 500 may be used, for example, to execute a method for

folding NOP data, such as the embodiments described herein. For purposes of this disclosure, a

15   processing system includes any system that has a processor, such as, for example; a digital signal

processor (DSP), a microcontroller, an application specific integrated circuit (ASIC), or a

microprocessor. System 500 is representative of processing systems based on the Itanium® and

Itanium® II microprocessors as well as the Pentium®, Pentium® Pro, Pentium® II, Pentium®

III, Pentium® 4 microprocessor, all of which are available from Intel Corporation. Other

20   systems (including personal computers (PCs) having other microprocessors, engineering

workstations, personal digital assistants and other hand-held devices, set-top boxes and the like)

may also be used. In one embodiment, system 500 may be executing a version of the Windows™

operating system available from Microsoft Corporation, although other operating systems and graphical user interfaces, for example, may also be used.

[00050]   Referring to Fig. 5, processing system 500 includes a memory system 502 and a processor 504 that includes folding logic 116 and a buffer 210 of the format shown in Fig. 2.

5   The buffer 210 may be implemented as a ROB (such as 112, Fig. 2). Memory system 502 may store instructions 510 and data 512 for controlling the operation of the processor 504.

[00051]   Memory system 502 is intended as a generalized representation of memory and may include a variety of forms of memory, such as a hard drive, CD-ROM, random access memory (RAM), dynamic random access memory (DRAM), static random access memory (SRAM), flash

10   memory and related circuitry. Memory system 502 may store instructions 510 and/or data 512 represented by data signals that may be executed by processor 504. The instructions 510 may include bundled instructions as discussed herein.

[00052]   In the preceding description, various aspects of a method and apparatus for NOP folding have been described. For purposes of explanation, specific numbers, examples, systems

15   and configurations were set forth in order to provide a more thorough understanding. However, it is apparent to one skilled in the art that the described method and apparatus may be practiced without the specific details. In other instances, well-known features were omitted or simplified in order not to obscure the method and apparatus.

[00053]   While particular embodiments of the present invention have been shown and

20   described, it will be obvious to those skilled in the art that changes and modifications can be made without departing from the present invention in its broader aspects. The appended claims are to encompass within their scope all such changes and modifications that fall within the true scope of the present invention.